2018 CPP Midterm Exam

Q1. Parking fee 15%

Please write a function to calculate the parking fee. The parking fee rules are listed as follows:

a. If the vehicle is a motorcycle, you need to pay NT$30 per hour for the first 3 hours, and if it parks over 3 hours, it will take NT$40 per hour.
b. If the vehicle is a car, you need to pay NT$50 for the first hour, you will spend more NT$10 every hour until the fee is reach NT$100.
c. If parking time is not integer, you need to round it up to an integer .
d. If vehicle is bicycle or bike, the fee is zero.
e. The vehicle name should allow first word uppercase or lowercase.
f. If the input vehicle not car, motorcycle, or bike or bicycle, print "System does not support this type of vehicle".
g. If the system supports neither the type of vehicle or "time number", then the priority of error message of vehicle type should be higher.

Input：

Each line of input contains a vehicle name and parking time.

Output:

Print the parking fee correspond to the input vehicle and parking time if the input is legal,
Or print the statement when system does not support this vehicle or system does not support this time number.

IO Example :

Sample Input	Sample Output
Motorcycle 4 motorcycle 5	The Parking fee is 130 The Parking fee is 170

motorcycle 5.5	The Parking fee is 210
Motor 10	System does not support this type of vehicle.
Motorcycle -60	System does not support this time number.
Car 2	The Parking fee is 110
Car 5	The Parking fee is 350
Car 5.5	The Parking fee is 450
Bike 100	The Parking fee is 0
Bicycle 10	The Parking fee is 0
Plane 50	System does not support this type of vehicle.
Iogfjias -60	System does not support this type of vehicle.

Q2. Big Numbers Multiplication 15%

題目說明： Compute the multiplication of very large integers.

a. Notice that the integer can be positive and negative.
b. The absolute value of the integer is smaller than 10^{19999}.

Input

Two integers separated with space.

Output

Please output the multiplication result in the form of a digit mode instead of a Scientific notation.

Notes:

a. Please include your class program in the test program by yourself, if you implement the function by class.
b. Your program must read input data until read EOF(Ctrl+z).
c. -0 is an illegal output and input.

IO範例：

Sample Input	Sample Output

-1 -1
0 100
-123456 7
987654 -321
	1
	0
	-864192
	-317036934

Q3. Parenthesis Checker 15%

Given an expression string **exp**, please examine whether the pairs and the orders of parenthesis, ‘{’, ‘}’, ‘(’, ‘)’, ‘[’, ‘]’, appear correctly in **exp**. For example, the program should print 'balanced' for **exp** = “[()]{}{[()()]()}” and 'not balanced' for **exp** = “[(])”.

Input:

a. The first line of input contains an integer T to denote the total number of test cases.
b. Each test case consists of a string of expression, in a separate line.
c. The characters of test will include only these following symbols‘{’, ‘}’, ‘(’, ‘)’, ‘[’, ‘]’.

Output:

Print 'balanced' without quotes if pair of parentheses are balanced, and otherwise, print 'not balanced' in a separate line.

Constraints:

$1 \leq T \leq 100$

$1 \leq ||\mathbf{exp}|| \leq 100$

Notes:

Example:

Sample Input	**Sample Output**
4	balanced

{([])} () ()[] [(])	balanced balanced not balanced

Q4. N-dimensional vector operation 15%

題目說明： Please implement a **VecNf** class for n-dimensional vector operations.

Please

1. Provide a default constructor and a parameterized constructor that enables an arbitrary vector to be constructed. The default constructor creates one-dimensional vector, the coefficient is zero. The parameterized constructor should construct an array data as vector data and the size of array.
2. Provide these operations:

vector = vector, (assign operation, assign a vector to another vector)
vector + vector, (plus operation)
vector - vector, (minus operation)
vector * vector, (return a number whose value is inner product),
vector * constant, (scale operation)
constant * vector, (scale operation)
vector[constant integer], (getter/setter operator, assign and extract values to the indexed dimension)

Before computing with two vector, make sure whether the dimensions are consistent. If the dimensions are inconsistent, you must print "dimensions inconsistent", then return the default vector or constant zero.

3. You should decide whether to implement these functions as members, friends, or standalone functions.

NOTES:

a. We will replace the **main** function with our program to test your program. Please don't implement any **VecNf** class program in **main** function.
b. The index is the indexed dimension to the vector. For example, the index of 0 points to the first dimension of the vector, the index of 1 point to the second dimension of the vector, etc.
c. You must include your **VecNf** class in the test program for our testing.

IO範例：

Sample Input

```
int main()
{
	VecNf empty;
	float xy[] = { 3.0, 2.0 };
	float xyz[] = { 1, 2, 3 };
	float xyz2[] = { 6, 5, 4 };

	VecNf vxy(xy, 2);
	VecNf vxyz(xyz, 3);
	VecNf vxyz2(xyz2, 3);

	VecNf t;
	t = vxy;		//test operator=
	t = vxyz;

	cout << "Vector xy " << endl;
	{for (int i = 0; i < 2; i++)
		cout << "dimension  " << i + 1 << " :  " << vxy[i] << endl;
	}
	cout << "Vector xyz " << endl;
	{for (int i = 0; i < 3; i++)
		cout << "dimension  " << i + 1 << " :  " << vxyz[i] << endl;
	}

	cout << "Vector add " << endl;
	VecNf add = vxyz + vxyz2;   //add test
```

Sample Output

```
Vector xy
dimension  1 :  3
dimension  2 :  2
Vector xyz
dimension  1 :  1
dimension  2 :  2
dimension  3 :  3
Vector add
dimension  1 :  7
dimension  2 :  7
dimension  3 :  7
dot = 28
dimensions inconsistent
dot = 0
```

```
        for (int i = 0; i < 3; i++)
                cout << "dimension  " << i + 1 << " :  " << add[i] << endl;

        float dot = vxyz * vxyz2;   //inner product test
        cout << "dot = " << dot << endl;

        float dot2 = vxy * vxyz;   //inner product test
        cout << "dot = " << dot2 << endl;

        system("pause");

        return 0;
}
```

Q5. Trapping Rain Water 20%

Given a sequence of N non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

For example:

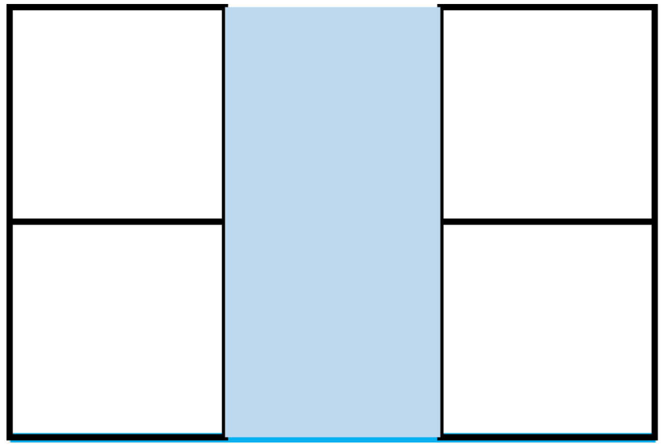
Input:

```
3
2 0 2
```

Output:

```
2
```

Structure is like below



We can trap 2 units of water in the middle gap.

Below is another example.

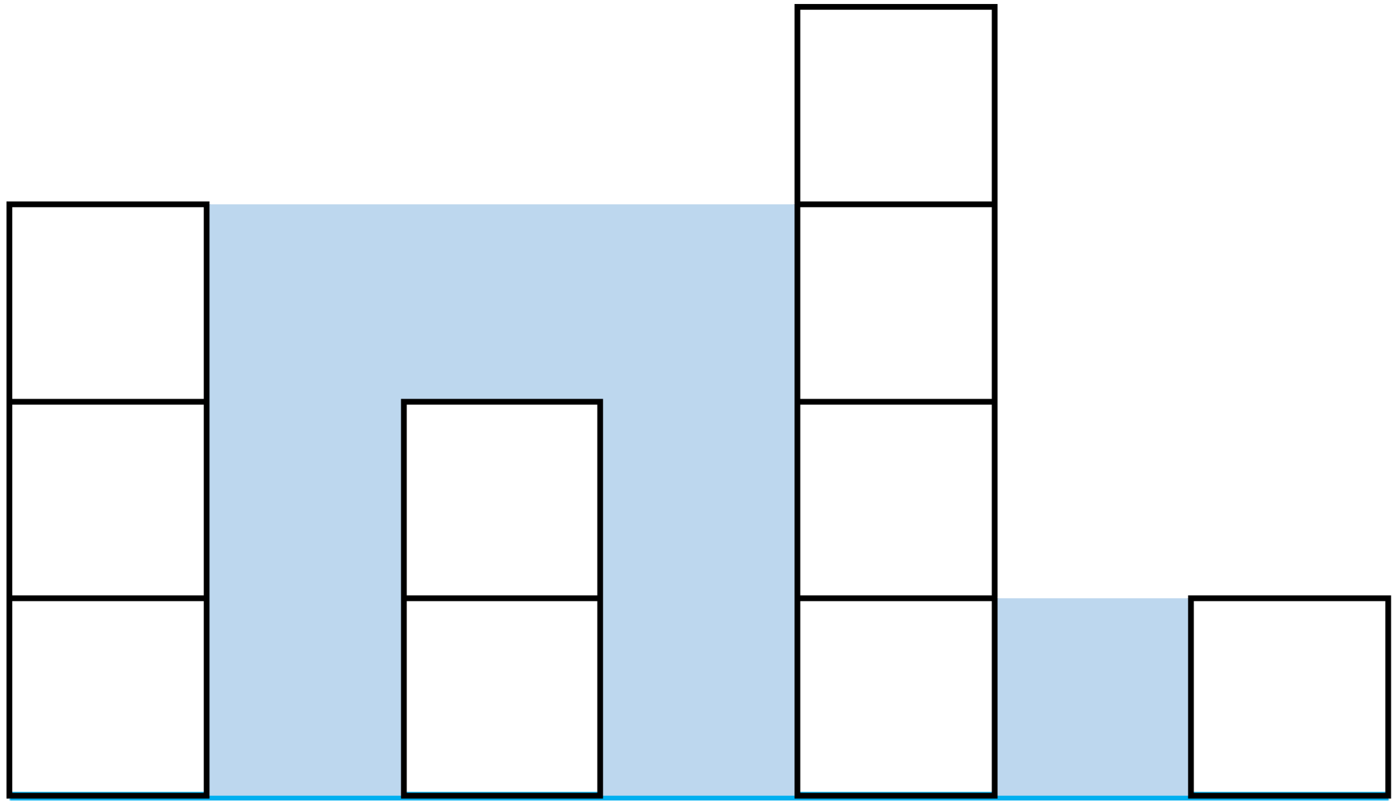
Input:

```
7
3 0 2 0 4 0 1
```

Output:

```
8
```

Structure is like below



We can trap 8 units of water in the middle gap.

Input:

The first line contains an integer **T** denoting the number of test cases and the following **T** lines should list out the description of the **T** test cases. The first number of each test case contain an integer **N** to specify the number of **Arr** and the following number **Arr[i]** specify the altitude of the **N** units.

Output:

Print the total trap units of water in the middle gap.

Constraints:

1<= T <=100

3<= N <=100

0<= Arr[i] <10

Note:

Example:

Sample Input	**Sample Output**

2
4 7 4 0 9
3 6 9 9
	10
	0

Q6. Football Game System 20%

Please write a football video game system. The system will take **SettingPlayer.txt** including all player information as input, and the system should read it in and analyze each line to create all corresponding players where each football player has PlayerID and two abilities: “Speed” and “Power”. The system can create a football player with **PlayerInformation** in the format of a **string**. The following describes the details.

a. The player information is a string composed of PlayerID, Name, Speed and Power in serialized form.
   Example: If the string is "05Chan5050" means PlayerID=5, Name=Chan, Speed=50, Power=50.
b. “Name” is a string of connected characters, whose length is 10 > length(name) > 2. “Speed” and “Power” are two digits to specify a value from 10 to 99.
c. The PlayerID is within the value from 01,02,03 ... to 10.
d. The shooting rate is defined by (Speed*0.5+Power*0.8) %, and set it to 100% if larger than 100%=
e. The ComparePlayer(string Player1 , string Player2) function compares two players’ Shooting rate and then prints the better one, such as “Player1 is the better player”. If two players have same Shooting rate, print “The two players have the same Shooting rate”
f. The Listplayer() function can list the all player by their Player ID from low to high.
g. If ComparePlayer(string Player1 , string Player2) calls the player who does not exist in the file, output "The player name Player1 does not exist" or "The player name Player1 and Player2 do not exist"
h. *If PlayerID in the **SettingPlayer.txt** repeats, the character and the abilities should be updated.
i. *The Name in the **SettingPlayer.txt** would not be repeated, so don't worry about that.
j. *If a PlayerID is not in the **SettingPlayer.txt**, PlayerInformation(ID) would print “The PlayerID ID has no character”

Please define a class of **FootballPlayer** with the following functions:
public:

```
    void SetFileName(string fileName);  // Read the file would not output anything
    void PlayerInformation(int ID);  // Output PlayerID, Name and Shooting rate
    void ComparePlayer(string Player1 , string Player2); //Output who is the better player or they have the same Shooting rate
    void ListPlayer(); // List all the player ID, player name and Shooting rate
```

Note：
Please do not change the function in main(). To test your program, we would use another file "SettingPlayer.txt". So make sure all the functions above can be executed and output correctly .
There is no space in the file "SettingPlayer.txt".
Each function output in main() function correspond to all Sample output.
IO範例 :

Sample Input	Sample Output

SettingPlayer.txt (File name)
01Orig1213
08Clark9595
01Arthur1213
03Bruce6050
04Zod9999
05Diana9060
-- 	
#include"FootballPlayer.h"
int main()
{
FootballPlayer footballPlayer1;
footballPlayer1.SetFileName("SettingPlayer.txt");
footballPlayer1.PlayerInformation(3);
footballPlayer1.PlayerInformation(4);
footballPlayer1.PlayerInformation(5);
footballPlayer1.PlayerInformation(6);
footballPlayer1.ComparePlayer("Clark", "Zod");
footballPlayer1.ComparePlayer("Clark", "Bruce");
footballPlayer1.ComparePlayer("Fury", "Stephen");
footballPlayer1.ComparePlayer("Clark", "Barry");
footballPlayer1.ComparePlayer("Tony", "Bruce");
	The PlayerID 3 is Bruce and Shooting rate:70%
	The PlayerID 4 is Zod and Shooting rate:100%
	The PlayerID 5 is Diana and Shooting rate:93%
	The PlayerID 6 has no character
	The two players have the same Shooting rate
	Clark is the better player
	The player name Fury and Stephen do not exist
	The player name Barry does not exist
	The player name Tony does not exist
	The player name Stephen and Steve do not exist
	The player name Vision and Steve do not exist

```
            footballPlayer1.ComparePlayer("Stephen", "Steve");
            footballPlayer1.ComparePlayer("Vision", "Steve");
            footballPlayer1.ListPlayer();
            system("pause");
      return 0;
}
```

```
All Player:
ID:01 Name: Arthur Shooting rate:16.4%
ID:03 Name: Bruce Shooting rate:70%
ID:04 Name: Zod Shooting rate:100%
ID:05 Name: Diana Shooting rate:93%
ID:08 Name: Clark Shooting rate:100%
```

```
SettingPlayer2.txt (File name)
05Peter8060
07Fury4545
01Tony3040
02Steve9090
03Vision7099
06Banner9999
08Stephen7060
------------------------------------------------------------------------------
#include"FootballPlayer.h"
int main()
{
            FootballPlayer footballPlayer1;
            footballPlayer1.SetFileName("SettingPlayer.txt");
            footballPlayer1.PlayerInformation(3);
            footballPlayer1.PlayerInformation(4);
            footballPlayer1.PlayerInformation(5);
            footballPlayer1.PlayerInformation(6);
            footballPlayer1.ComparePlayer("Clark", "Zod");
            footballPlayer1.ComparePlayer("Clark", "Bruce");
            footballPlayer1.ComparePlayer("Fury", "Stephen");
            footballPlayer1.ComparePlayer("Clark", "Barry");
            footballPlayer1.ComparePlayer("Tony", "Bruce");
            footballPlayer1.ComparePlayer("Stephen", "Steve");
            footballPlayer1.ComparePlayer("Vision", "Steve");
            footballPlayer1.ListPlayer();
            system("pause");
```

```
The PlayerID 3 is Vision and Shooting rate:100%
The PlayerID 4 has no character
The PlayerID 5 is Peter and Shooting rate:88%
The PlayerID 6 is Banner and Shooting rate:100%
The player name Clark and Zod do not exist
The player name Clark and Bruce do not exist
Stephen is the better player
The player name Clark and Barry do not exist
The player name Bruce does not exist
Steve is the better player
The two players have the same Shooting rate
All Player:
ID:01 Name: Tony Shooting rate:47%
ID:02 Name: Steve Shooting rate:100%
ID:03 Name: Vision Shooting rate:100%
ID:05 Name: Peter Shooting rate:88%
ID:06 Name: Banner Shooting rate:100%
ID:07 Name: Fury Shooting rate:58.5%
```

```
    return 0;
}
```
	ID:08 Name: Stephen Shooting rate:83%